



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No. : 10/690,783 Confirmation No. 5629
Applicant : David J. Monnie et al.
Filed : October 21, 2003
TC/A.U. : 2194
Examiner : Seye, Abdou K.
Docket No. : 8474.0005
Customer No. : 00152
Title: : OBJECT MONITORING SYSTEM IN SHARED OBJECT SPACE

Mail Stop Amendment

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

COVER LETTER AND ACCOMPANYING DECLARATION

Sir:

The accompanying declaration starting on page 2 authenticates the attached evidentiary documents, which documents are herein submitted in support of the amendment being filed concurrently herewith. It is respectfully requested that the examiner consider these documents and, insofar as they represent information possibly subject to later revision, to include them in the record. This declaration comports with the requirements of 37 C.F.R. §§ 1.68 and 1.132 and also with official Office practice as set forth in the MPEP § 609.05(c).

DECLARATION UNDER 37 C.F.R. § 1.132 AUTHENTICATING DOCUMENTS

I, Kevin L. Russell, the undersigned representative for applicant, hereby declare that to the best of my knowledge, information, and belief, the attached exhibits, aside from handwritten notations identifying separate exhibits and marking portions of interest, are genuine copies of documents electronically accessed over the Internet from web-based sources and that such exhibits represent information that originated independently of the present applicant. It is my belief that these documents reflect how certain terms used in the claims have acquired a recognized meaning in the art. These documents comprise the following list of exhibits A-F:

Exhibit A) Wikipedia article entitled "Java Virtual Machine" posted at http://en.wikipedia.org/wiki/Java_virtual_machine and accessed February 2, 2007 (4 pp.);

Exhibit B) Wikipedia article entitled "Kaffe" posted at <http://en.wikipedia.org/wiki/Kaffe> and accessed February 4, 2007 (1 p.);

Exhibit C) Wikipedia article entitled "List of Java virtual machines" posted at http://en.wikipedia.org/wiki/List_of_Java_virtual_machines and accessed February 4, 2007 (2 pp.);

Exhibit D) Sun Microsystems, Inc. FAQ article (selected portions) entitled "Free and Open Source Java" posted at www.sun.com/software/opensource/java/faq.jsp and accessed February 2, 2007 (12 pp.);

Exhibit E) Sun Microsystems, Inc. article entitled "Sun Opens Java" posted at www.sun.com/2006-1113/feature/story.jsp and accessed February 2, 2007 (2 pp.); and

Exhibit F) Wikipedia article entitled "Virtual machine" posted at http://en.wikipedia.org/wiki/Virtual_machine and accessed February 4, 2007 (6 pp.).

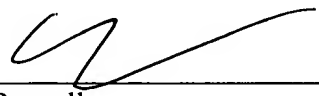
I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements, and the like so made, are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States

Appl. No. 10/690,783
Cover Letter and Declaration filed Feb. 21, 2007
Reply to Office Action of Dec. 12, 2006

Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Signed this 21st day of February, 2007

By



Kevin L. Russell
Of Attorneys for Applicant
Reg. No. 38,292
1600 ODS Tower
601 SW Second Avenue
Portland, OR 97204
Tel: (503) 227-5631

Certificate Of Mailing

I hereby certify that this declaration is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on February 21, 2007.

Dated: February 21, 2007


Kevin L. Russell



Java Virtual Machine

From Wikipedia, the free encyclopedia
(Redirected from Java virtual machine)

A **Java Virtual Machine (JVM)**, is a **virtual machine** that interprets and executes Java bytecode. This code is most often generated by Java language compilers, although the JVM can also be targeted by compilers of other languages. JVM's may be developed by other companies as long as they adhere to the JVM standard published by Sun.

The JVM is a crucial component of the Java Platform. The availability of JVMs on many types of hardware and software platforms enables Java to function both as middleware and a platform in its own right. Hence the expression "Write once, run anywhere." The use of the same bytecode for all platforms allow Java to be described as "Compile once, run anywhere", as opposed to "Write once, compile anywhere", which describes cross-platform compiled languages.

Starting with J2SE 5.0, changes to the JVM specification have been developed under the Java Community Process as JSR 924^[1]. As of 2006, changes to specification to support changes proposed to the class file format (JSR 202^[2]) are being done as a maintenance release of JSR 924. The specification for the JVM is published in book form^[3], known as "blue book". The preface states:

We intend that this specification should sufficiently document the Java Virtual Machine to make possible compatible clean-room implementations. Sun provides tests which verify the proper operation of implementations of the Java Virtual Machine.

The Sun implementation is called HotSpot. An example of a clean-room Java implementation could be Kaffe. Sun retains control over the Java trademark, which it uses to certify implementation suites as fully compatible with Sun's specification.

Contents

- 1 Execution environment
 - 1.1 Bytecode verifier
 - 1.2 Bytecodes
 - 1.3 Secure execution of remote code
- 2 References
- 3 See also
- 4 External links

Execution environment

Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of .class files. A program may consist of many classes, in which case every class will be in a different file. For easier distribution of large programs, multiple class files may be packaged together in a .jar file.

Execution of .class or .jar files is then handled by the JVM runtime which carries out emulation of the JVM



instruction set by interpreting it or by applying a just-in-time compiler (JIT) such as Sun's HotSpot. JIT compilation is used (rather than interpreters) in most JVM's today to achieve greater speed.

The Java Virtual Machine has a stack based architecture. In contrast, JIT compilers usually compile the byte code into register based machine code. Each thread has its own stack and program counter.

Bytecode verifier

The JVM *verifies* all bytecode before it is executed. This means that only a limited amount of bytecode sequences form valid programs, e.g. a JUMP (branch) instruction can only target an instruction within the same function. Because of this, the fact that JVM is a stack architecture does not imply a speed penalty for emulation on register based architectures when using a JIT compiler: in the face of the code-verified JVM architecture, it makes no difference to a JIT compiler whether it gets named imaginary registers or imaginary stack positions that need to be allocated to the target architecture's registers. In fact, code verification makes the JVM different from a classic stack architecture whose efficient emulation with a JIT compiler is more complicated and typically carried out by a slower interpreter.

Code verification also ensures that arbitrary bit patterns cannot get used as an address. Memory protection is achieved without the need for a MMU. Thus, JVM is an efficient way of getting memory protection on simple architectures that lack a MMU.

Bytecodes

The JVM has instructions for the following groups of task

- Load and store
- Arithmetic
- Type conversion
- Object creation and manipulation
- Operand stack management (push / pop)
- Control transfer (branching)
- Method invocation and return
- Throwing exceptions
- Monitor-based concurrency

The aim is binary compatibility. Each particular host operating system needs its own implementation of the JVM and runtime. These JVMs interpret the byte code semantically the same way, but the actual implementation may be different. More complicated than just the emulation of bytecode is compatible and efficient implementation of the Java core API which has to be mapped to each host operating system.

Secure execution of remote code

A virtual machine architecture allows very fine-grained control over the actions that code within the machine is permitted to take. This is designed to allow safe execution of untrusted code from remote sources, a model used by Java applets. Applets run within a VM incorporated into a user's browser, executing code downloaded from a remote HTTP server. The remote code runs in a restricted "sandbox", which is designed to protect the user from misbehaving or malicious code. Publishers can purchase a certificate with which to digitally sign applets as "safe", giving them permission to break out of the sandbox and access the local file system and network.

References

1. ^ JSR 924 – Specifies changes to the JVM specification starting with J2SE 5.0
2. ^ JSR 202 – Specifies a number of changes to the class file format
3. ^ *The Java Virtual Machine Specification* (the first and second editions are also available online)
1. *Clarifications and Amendments to the Java Virtual Machine Specification, Second Edition* includes list of changes to be made to support J2SE 5.0 and JSR 45
2. JSR 45 – Specifies changes to the class file format to support source-level debugging of languages such as JSP and SQLJ that are translated to Java

See also

- List of Java virtual machines
- Parrot virtual machine
- Common Language Runtime
- C to Java Virtual Machine compilers
- Java Runtime Environment

External links

- List of languages which compile to the Java virtual machine.
- Kaffe.org - the Kaffe project
- JamVM - The Jam Virtual Machine
- The lean, mean, virtual machine - An introduction to the basic structure and functionality of the Java Virtual Machine
- Java Glossary - installing Java useful tips for installing Java for users and developers
- Test your Java Virtual Machine

Java (Sun)

Major Java (programming language) | Java Platform | Java Development Kit | **Java Virtual Technologies: Machine** | Java Runtime Environment

History: Java version history | Criticism of Java | Java Community Process | Sun Microsystems

Language Bytecode | Syntax | Applets | Servlets | JavaServer Pages

Features:

Sun Microsystems

[hide]

Software: Solaris • StarOffice/OpenOffice.org • Java Desktop System • Java (Java language • **JVM** • Java API) • JES • Network File System

Hardware: SPARCstation • Sun Ultra series • Sun Enterprise • Sun Blade • Sun Fire • UltraSPARC T1 • SPARC • JavaStation • Sun Ray

Education and Recognition: SCPs

Retrieved from "http://en.wikipedia.org/wiki/Java_Virtual_Machine"

Categories: Java platform | Java programming language | Java virtual machine | Java specification requests | Virtual

machines

- This page was last modified 09:39, 25 January 2007.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.

Kaffe

From Wikipedia, the free encyclopedia

Kaffe is a clean room design of a Java Virtual Machine. It comes with a subset of the Java 2 Platform, Standard Edition Java API and tools needed to provide a Java runtime environment. Like most other Free Java virtual machines, Kaffe uses GNU Classpath as its class library.

Kaffe is a lean and portable virtual machine, although it's significantly slower than commercial implementations^[1]. When compared to the reference implementation of the JVM written by Sun Microsystems, Kaffe is significantly smaller; it thus appeals to embedded system developers. It comes with just-in-time compilers for many of the CPU architectures, and has been ported to more than 70 system platforms in total. It runs on devices ranging from embedded SuperH devices to IBM zSeries mainframe computers. it will even run on a PlayStation 2.

Kaffe is free software, licensed under the terms of the GNU General Public License (GPL). It is being developed by a world-wide team of programmers. Beside the mailing list, the developers can often be reached via IRC in the #kaffe channel on irc.freenode.org.


Kaffe is also the Danish, Norwegian and Swedish word for *coffee*. The name was probably chosen because Java is the name of a coffee bean.

References

- ↑ http://www.shudo.net/jit/perf/

See also

- List of Java virtual machines
- GNU Classpath

 *Free software Portal*

Java Virtual Machines

[hide]

Sun HotSpot • Apache Harmony • BEA JRockit • GCJ • Squawk • IKVM • **Kaffe** • SableVM • CACAO • ElectricalFire • JikesRVM • JNode • JamVM • *more ...*

Retrieved from "http://en.wikipedia.org/wiki/Kaffe"

Categories: Java virtual machine | Free compilers and interpreters

- This page was last modified 00:35, 20 January 2007.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.

List of Java virtual machines

From Wikipedia, the free encyclopedia

The following is a non-exhaustive list of J2SE **Java virtual machines**. There are a large number of Java Platform, Micro Edition (J2ME) vendors not listed here. Note that J2EE runs on the standard J2SE JVM but that some vendors specialize in providing a modified JVM optimized for J2EE applications. A large amount of Java development work is done on either Windows or Linux, and that is primarily done with the Sun JVM, which is usually considered to be adequate quality for commercial deployment. The picture is even further complicated by 32-bit/64-bit varieties.

Contents

- 1 Proprietary/closed source implementations
 - 1.1 Lesser known proprietary JVMs
- 2 Free/open source implementations
- 3 See also

Proprietary/closed source implementations

- Hewlett-Packard's Java for HP-UX, OpenVMS, Tru64 and Reliant(Tandem) UNIX UNIX platforms [1]
- IBM for MVS, AIX, OS/400, z/OS [2]
- Apple Computer MacOS Runtime for Java (MRJ) [3]
- BEA Systems JRockit [4]

Lesser known proprietary JVMs

- Squawk virtual machine (Sun JVM for embedded system and small devices)
- Blackdown Java [5] (port of Sun JVM)
- Excelsior JET [6] (with AOT compiler)
- Gemstone Gemfire JVM - modified for J2EE features
- Golden Code Development [7] (EComStation and OS/2 port of Java RTE and SDK for J2SE v1.4.1_07)
- Novell, Inc. [8]
- TAO-Groups Elate/Intent [9]
- NSIcom CrE-ME [10]

Free/open source implementations

- AegisVM [11]
- Apache Harmony [12]
- CACAO [13]
- GCJ [14]
- IKVM.NET [15]
- Jamiga [16]
- JamVM [17]
- Jaos [18]
- JC [19]

- Jikes RVM [20]
- JNode [21]
- JOP [22]
- Jupiter JVM [23]
- Kaffe [24]
- leJOS [25]
- NanoVM [26]
- SableVM [27]
- Sun Microsystems' Java HotSpot Virtual Machine [28]
- Waba [29]
- Wonka [30]

See also

- Free Java implementations

Java Virtual Machines

[hide]

Sun HotSpot • Apache Harmony • BEA JRockit • GCJ • Squawk • IKVM • Kaffe • SableVM • CACAO • ElectricalFire • JikesRVM • JNode • JamVM • *more ...*

Retrieved from "http://en.wikipedia.org/wiki/List_of_Java_virtual_machines"

Categories: Java virtual machine | Lists of software

- This page was last modified 17:57, 16 January 2007.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.

Java Solaris Communities About Sun How to Buy My Account Cart

United States Worldwide

Home > Products > Software >

Free and Open Source Java



Free Range.

Check the terrain for new opportunities.



Overview About Java Community FAQ Get Involved



FAQ

- 1 The Announcement
- 2 Java SE (JDK) Announcement Specifics
- 3 Java ME Announcement Specifics
- 4 Java EE and NetBeans Announcement Specifics
- 5 General and Goals
- 6 Business Model
- 7 License
- 8 Code and Encumbrances
- 9 Governance
- 10 The Java Brand
- 11 TCKs
- 12 JCP
- 13 Community Development and Infrastructure
- 14 Open-Source Communities and Java

→ The Announcement

- What is Sun announcing?
- What is the significance of this announcement to the industry?
- Why did you choose the GPL for Java? Why not the CDDL that Sun created for OpenSolaris? Why not the Apache Software License?

Q: What is Sun announcing?

A: Sun is open sourcing all of its Java platform implementations under same license (called GPL Version 2) used by the GNU/Linux operating system

Specifically Sun is announcing

- GPL v2 license for Sun's Java SE (JDK) and Java ME implementations, and adding this license to Sun's Java EE implementation
- First release of code for the JDK and for Sun's Java ME implementation, projects and

Source

OpenJDK
Mobile & Embedded
GlassFish

Communities

OpenJDK
Mobile & Embedded
GlassFish

Related Resources

Developers sun.com
NetBeans.org
Java.net
Java.sun.com
Java.com
Java Community Process



- communities
- Roadmap for future code releases and community development

This singular act is the largest contribution ever made to the free software community, and places Sun squarely at the front of the open-source movement - as the single biggest commercial contributor

: Back to top

Q: What is the significance of this announcement to the industry?

A: This announcement celebrates the open sourcing of the code base for one of the industry's most significant and pervasive software platforms, to foster adoption in new markets, to build broader communities, and fuel even more innovation. With over 3.8 billion Java technology enabled devices, Java technology has already demonstrated explosive growth, appearing in volume nearly everywhere. Now, as free software, the Java platform can address new markets and be the engine of innovation for the next generation of networked applications.

: Back to top

Q: Why did you choose the GPL for Java? Why not the CDDL that Sun created for OpenSolaris? Why not the Apache Software License?

A: One of our primary goals in this move was to grow the Java market, benefiting everyone - platforms, ISVs, OEMs, customers of every type as the reach and richness of the Java market grows. We considered a number of different license choices, including CDDL. While each of the open-source licenses we considered had a positive potential impact, our evaluation brought us to the conclusion that using the GPL would result in the greatest incremental growth to the Java market.

: Back to top

→ Java SE (JDK) Announcement Specifics

- What is the OpenJDK Community?
- What components of the JDK software are you open sourcing today?
- Is the JDK source code under the GPL as well?
- When will you finish open sourcing the JDK? What is the timeline?
- Will Sun continue distributing its commercial implementation of the JDK?
- Will Sun's commercial JDK releases be built from the open-source code?
- What components of the JDK software are you open sourcing today? Why did you choose these components?
- Are you open sourcing the Java language or the Java SE platform specifications?
- Which version of the JDK do these components come from? Are you open sourcing the latest code?
- What is the Java compiler ("javac")? What can developers do with this code?
- What is the Java HotSpot virtual machine? What can developers do with this component?
- What is JavaHelp? What is the relationship of JavaHelp to the JDK?
- Where do developers go to participate in implementing these JDK components?
- How can I get involved?
- What is the governance model for the OpenJDK projects?
- How does the governance and infrastructure of these first open-source JDK projects differ from the existing JDK Community and jdk6 and jdk7 projects on java.net?

Q: What is the OpenJDK Community?

A: Sun is establishing the OpenJDK Community for the ongoing development of Sun's open-source implementation of Java SE. The OpenJDK Community is where developers will gather to collaborate on the open-source JDK code base and related projects. The OpenJDK project in which the open-source code base will live is part of this community. Through the OpenJDK project, developers will be able to directly influence the future of the JDK implementation, participate with their peers in an open community and help take Java technology where it hasn't been before. Sun is evolving the existing JDK Community, where developers have been working on the source code over the past two years, into a site where Sun and non-Sun developers alike can collaborate together on the implementation. The OpenJDK Community can be found at

| <http://community.java.net/openjdk>

: Back to top

Q: What components of the JDK software are you open sourcing today?

A: We're open sourcing the Java programming language compiler ("javac"), and the Java HotSpot virtual machine. In addition, we're open sourcing the JavaHelp 2.0 extensible help system, Sun's reference implementation of JSR 97 in this first code release. The JavaHelp code base is found at <https://javahelp.dev.java.net>

: Back to top

Q: Is the JDK source code under the GPL as well?

A: Yes, the JDK is licensed under the GPL version 2 with the Classpath exception. This license, which will be used for all of Sun's Java implementations, is endorsed by the Free Software Foundation, and is the license used by the GNU/Linux operating system.

: Back to top

Q: When will you finish open sourcing the JDK? What is the timeline?

A: We expect to release a fully buildable JDK based almost completely on open-sourced code in the first half of 2007.

: Back to top

Q: Will Sun continue distributing its commercial implementation of the JDK?

A: Yes. For people who want the benefits of commercial support, and predictability, they may choose to use Sun's commercial distribution of the JDK or JRE. The free commercial implementation of the JDK may be found at <http://java.sun.com/javase/downloads/index.jsp>. Similarly, Sun's free JRE may be downloaded from <http://java.com>. To learn more about development and deployment support options, visit <http://sun.com/javasupport>.

: Back to top

Q: Will Sun's commercial JDK releases be built from the open-source code?

A: Yes, for the most part. Since there's some encumbered code in the JDK, Sun will continue to use that code in commercial releases until it's replaced by fully-functional open-source alternatives.

: Back to top

Q: What components of the JDK software are you open sourcing today? Why did you choose these components?

A: We're open sourcing the Java programming language compiler ("javac"), and the Java HotSpot virtual machine. In addition, we're open sourcing the JavaHelp 2.0 extensible help system, Sun's reference implementation of JSR 97 in this first code release. The JavaHelp code base is found at <https://javahelp.dev.java.net>. Sun will release all of the JDK as noted in the next few quarters as we work through encumbrance issues.

Fortunately, we have these major subcomponents of the JDK that are both unencumbered and that embody some of the most significant innovations in Java technology.

: Back to top

Q: Are you open sourcing the Java language or the Java SE platform specifications?

A: We are not open sourcing the Java programming language, nor the platform APIs and specifications, which are governed by the JCP. We're open sourcing Sun's implementations of the Java SE and Java ME specifications.

: Back to top

Which version of the JDK do these components come from? Are you open sourcing the latest code?

A: We're open-sourcing these components from a very early build of JDK 7. In order to prepare these components to be open sourced, we not only changed the license text but we also simplified the build process in order to make these components more easily buildable outside of the full JDK source tree. JDK 6, is nearly finished, hence we're releasing these components from the JDK 7 tree. The only other differences between the JDK 6 and 7 versions of these components are minor bug fixes and enhancements that have already been integrated into the JDK 7 tree. When we open-source the full JDK we'll make the sources for both JDK 6 and JDK 7 available. The community will have both a stable release - JDK 6 - on which to focus quality improvements, and JDK 7, the next feature release where all the action will be for innovation and new capabilities.

⌕ Back to top

Q: What is the Java compiler ("javac")? What can developers do with this code?

A: javac is the compiler that translates Java programs into "byte codes" that are then executed by a Java virtual machine (JVM). Developers will be able to experiment with the compiler, fix bugs, and try out new optimizations with this early code drop.

⌕ Back to top

Q: What is the Java HotSpot virtual machine? What can developers do with this component?

A: Java HotSpot is Sun's implementation of the Java virtual machine. The Java HotSpot VM includes the core execution engine for the Java platform, including

- dynamic compilers that convert Java bytecodes into optimized native machine code on supported hardware platforms
- memory management and garbage collection subsystems
- threads and synchronization
- monitoring, debugging, and profiling telemetry
- parts of the Java security architecture including the bytecode verifier

As such it is the component of the Java SE platform that delivers most directly on the "Write Once, Run Anywhere" promise of Java technology.

Developers will be able to learn how a world-class virtual machine is built, fix bugs, experiment with new garbage collection, synchronization, and bytecode compiler algorithms, and port the VM to new hardware architectures and operating systems with this code.

⌕ Back to top

Q: What is JavaHelp? What is the relationship of JavaHelp to the JDK?

A: JavaHelp software is a full-featured, platform-independent extensible help system that lets you incorporate online help in applets, components, applications, operating systems, and devices. Authors can also use JavaHelp software to deliver online documentation for the Web, and for corporate intranets. JavaHelp is Sun's reference implementation for JSR 97.

While JavaHelp is not part of the JDK, it is a closely related technology, and is being open sourced at the same time as the first code components of the JDK.

⌕ Back to top

Q: Where do developers go to participate in implementing these JDK components?

A: Developers can gain immediate access to these components under the open-source license by visiting <https://openjdk.dev/java.net/>

⌕ Back to top

Q: How can I get involved?

A: Visit the OpenJDK Community at <http://community.java.net/openjdk> and check out the projects.

In addition, the NetBeans Mobility Pack makes it easy to create mobile applications with drag-and-drop screen design. The world-record producing Sun Studio development environment lets you work on the platform-specific native code in the Java HotSpot virtual machine.

⌕ Back to top

General and Goals

- Why is Sun open sourcing its Java implementations now?
- What influenced this decision?
- Why is this good for Java developers?
- How does this benefit Java customers and users?
- What markets do you believe this initiative will open up for the JDK?
- What impact will this move have on the adoption of the Java platform?
- What are your goals in open sourcing Sun's Java ME implementations?
- Who benefits from open sourcing Java ME?
- What does Sun mean by "compatibility?"
- Do you think anyone will fork the JDK?
- So what about compatibility? How will Java technology remain "Write Once Run Anywhere" after Sun's Java platform implementations are open sourced?

Q: Why is Sun open sourcing its Java implementations now?

A: The Java platform is 11+ years in the making. When it was first released it was a radical departure for commercial software, including full source code under a novel license. Sun and the Java ecosystem have been extremely successful at establishing and growing a very large and dynamic market in this time. The Java platform retained its own licensing model even as the open-source model proliferated, because the Java licensing model successfully created a large and open market with many compatible choices. We now see an opportunity to encourage even more possibilities for adoption in places the Java platform hasn't gone before, where Free licensing and open-source development approaches are a prerequisite for consideration.

At the same time, the Free software and open-source communities are now saying that compatibility is a given for any Java implementation. And there is a new spirit of innovation with Web 2.0, SOA and collaboration/participation technologies, and the Java platform is the perfect foundation platform on which to innovate - and open-source can help accelerate this innovation.

⌕ Back to top

Q: What influenced this decision?

A: Key Free software and open-source communities have stated that they believe that only Java technology implementation that are completely compatible with the specification can succeed in the market. These communities, which provide thought leadership for the open-source Java world, are now focused on delivering only compatible implementations.

In addition, Sun has gained experience in community development with the Project GlassFish open-source implementation of Java EE, with OpenSolaris, with OpenOffice.org and NetBeans, and with the JDK Community on java.net. This experience makes us confident that when we open-source Sun's Java implementations, the platform will benefit, and we can better balance the needs of community with those of customers, end users, and licensees.

Overall, we feel that caution is appropriate for a technology that affects the lives and livelihoods of tens of millions of people around the world. After much reflection we feel now is the perfect time to take the next step with the Java platform.

⌕ Back to top

Q: Why is this good for Java developers?

A: Because volume wins. Open sourcing Sun's Java SE implementation will lower barriers to adoption in markets where open-source software leads. As new markets adopt Java technology, developers will discover new opportunities. More applications. Innovations that leverage the industrial strength foundation of Java to deliver valuable new products and services. And

developers will be able to directly influence the future of the JDK implementation, participating with their peers in an open community. Taking Java where it hasn't been before, and helping to ensure that Java technology remains a central unifying standard for the Internet.

Back to top

Q: How does this benefit Java customers and users?

A: Your investment is safe, with multiple independent implementations of Java SE, and the reference implementation from Sun available under an open-source license. An open-source JDK provides peace of mind, plain and simple.

- You can adopt Sun's Java technologies with full confidence. Java SE will be freely available - subject to the market forces that drive competition, reduce price and speed innovation.
- Whether you value Free and open-source software for philosophical or economic reasons, want more flexibility, appreciate the quality that transparency brings, or want to know you have ultimate control over your investment, you can adopt the JDK knowing that you'll gain these advantages.
- With the rock-solid, high performance JDK implementation from Sun under the same open-source license used by GNU/Linux distributions, you can expect Java technology to find its way onto new platforms, be leveraged for new applications and infrastructure, and be the foundation of tomorrow's most exciting new products and web technologies.

Your investment in Java technology will become even more valuable as open-source speeds innovation, spreads adoption, and carries the platform to places it couldn't previously go.

Back to top

Q: What markets do you believe this initiative will open up for the JDK?

A: Several markets are likely to find the JDK to be more suitable for use once it is under an open-source license.

- Government, educational, and research markets where open-source software is either mandated, or highly desired.
- Enterprises that mandate open-source infrastructure to retain maximum flexibility and drive competitive procurement.
- Enterprises and organizations that have selected OS distributions based on GNU/Linux and OpenSolaris as preferred OSs for deployment.

In order to gain the benefits of commercial support, and predictability, these customers may choose to use Sun's commercial distribution of the JDK or JRE in conjunction with a support contract. However, they would not consider the commercial product if the open-source version were not available.

Back to top

Q: What impact will this move have on the adoption of the Java platform?

A: The Java platform has been very widely adopted already - it is one of the most important and widely used components of modern web-based infrastructure. But there remain un-tapped or under-served markets. In particular, Java technology is not always included in open-source web infrastructure ÖstacksÖ that are commonly distributed and deployed alongside and included with GNU/Linux distributions. Those that do include Java technology often do not include an up-to-date, compatible runtime and development environment. We will be working with the GNU/Linux distributions to get the JDK included as part of the free software repositories commonly included with these open-source operating system distros. Once the JDK is easily obtained and installed with these platforms, we expect to see more widespread adoption of Java technology outside of North America and Western Europe, as well as in cutting edge web infrastructure deployments worldwide.

Since the GPL is a very widely used open-source license (in fact it's the same license used by GNU/Linux), distributing the JDK under the GPL with GNU/Linux distributions should be a good match, making it easier to adopt by those looking for open-source alternatives.

Back to top

Q: What are your goals in open sourcing Sun's Java ME implementations?

A: To remain the number one mobile application development platform*. Java ME needs to grow and evolve. This means engaged developers, accelerated innovation, and a more consistent implementation across devices [* Evans Data Corp Spring 2006]

: Back to top

Q: Who benefits from open sourcing Java ME?

A: Everyone in the Java ME ecosystem benefits from open sourcing Java ME. Benefits include

- simplified evaluations,
- transparent development,
- opportunities to directly influence the future of the platform

By open sourcing these implementations, handset manufacturers can leverage the common code base to reduce their development costs. The result will be less variation across devices & in other words, "Write Once, Run Anywhere" taken to the next level.

Both application developers and operators will benefit from the accelerated innovation, enabling developers to continue to create compelling applications and services, which in turn help drive revenue. Operators and handset manufacturers will benefit from lower porting, testing and maintenance costs.

: Back to top

Q: What does Sun mean by "compatibility?"

A: Compatibility for a Java technology means the implementation of that technology meets the associated compatibility requirements of its Technology Compatibility Kit or TCK. For Java SE this means passing the TCK tests and other requirements defined in the Java SE TCK.

: Back to top

Q: Do you think anyone will fork the JDK?

A: We expect great new ideas and valuable research to come from forks to the platform. In addition, Sun encourages compatible forks where the code is ported to additional hardware and software platforms. Such ports extend the breadth of Java SE to places currently not supported by any vendor. Broad distribution of incompatible forks is potentially a danger since such forks could damage the "Write Once, Run Anywhere" compatibility value of the Java platform.

: Back to top

Q: So, what about compatibility? How will Java technology remain "Write Once, Run Anywhere" after Sun's Java platform implementations are open sourced?

A: The Java technology compatibility promise is so central to the value of the platform that it has been the single most important influence in driving the detailed planning and decisions for this initiative. Sun is making a number of key decisions and commitments to the community that we believe will help foster compatibility.

- License: GPL makes proprietary forks less likely, as all changes must be published
- Branding: the Java trademark and logos are for compatible implementations only
- JCP: the JCP's role as the governing body for Java standards and evolution is not changed by this move
- Community Development: Sun's experience in building strong communities means developers will likely find the governance and infrastructure to their liking

In addition to the specific steps that we're taking to help foster compatibility, we are convinced that the market will demand compatible implementations, and that incompatible ones won't gain traction. With the billions of dollars of Java applications in the installed base, the community has recognized that a Java implementation that doesn't run the installed base of code won't get very far. That is why both the GNU/Classpath and Apache Harmony projects have stated unequivocally that they're working to build fully compatible implementations of Java SE.

from license section

Q: I want to contribute. Do I need to sign anything to get started?

A: Yes Sun requires that contributors to all of its Free and open-source projects sign the Sun Contributor Agreement (SCA) and mail or fax back the completed agreement. A copy of the SCA can be found at https://jdk.dev/java/net/Sun_Contributor_Agreement_v1.2.pdf

Back to top

Q: Why do you have a Contribution Agreement?

A: The Sun Contributor Agreement (SCA) is needed to ensure that Sun has the rights to use your contributions in products and projects. It also asks you to warrant that you are entitled to grant Sun these rights, and that, to your knowledge, your contribution does not violate anyone else's rights. Sun's responsibilities to existing licensees are not removed by Freeing these implementations, so Sun needs this sharing in order to serve the extended Java community.

Back to top

Q: Do I lose any rights to my contribution under the SCA?

A: Unlike some other contribution agreements that require you to transfer copyrights to an organization, the SCA does not take away any of your rights to your contributed intellectual property. When you agree to the SCA, you grant Sun joint ownership in copyright, and a patent license for your contributions. You retain all rights, title, and interest in your contributions and may use them for any purpose you wish.

Back to top

Q: But other communities are non-profits. Why should I share my copyrights with Sun?

A: Sharing copyrights protects Java Community interests. Sun acts as a bridge, ensuring that both open-source communities and traditional commercial organizations are able to grow the Java ecosystem, for the benefit of all. To do this, Sun needs to be able to offer the Java platform as a whole under both open-source and commercial licenses. The SCA enables Sun to do that. Additionally it allows Sun to ensure that the origins of every line of code are known, in case of future litigation against Sun or the community, and it allows Sun to upgrade the open-source licenses in the future should that become necessary.

Back to top

Q: What can Sun do with my contribution?

A: Sun may exercise all rights that a copyright holder has, as well as the rights you grant in the SCA to use any patents you have in your contributions. So may you.

Back to top

Q: Will Sun continue to offer the JDK and Java ME source code under a commercial license, along with GPL v2?

A: Yes. Indeed, some of Sun's existing licensees will continue to prefer a commercial license over an open-source license for a variety of reasons.

→ The Java Brand

- What is the Java brand?
- Can I call products I create from code I download from your open-source site "Java"?
- Can I use the term "JDK" to describe my product - it's built with code from the OpenJDK Project after all?
- How do I know if someone else's product is compatible?
- How can I describe a product I create using code from the open-source site if I don't wish to get it certified by Sun?
- What is the Java Verified Program?

Q: What is the Java brand?

A: The Java brand has two components: its name, Java, and its logo, the Java cup and steam. In addition, there are "sub-brands" that further clarify your use of Java technology and define how a product complies with its brand promise: Java Powered and Java Compatible.

These brands - and their logos - are valued registered trademarks owned, protected, and managed by Sun Microsystems. The Java brand carries with it a very specific value proposition that tech-savvy consumers and IT professionals everywhere recognize - a promise of "Write Once, Run Anywhere", and an expectation of exciting, robust, easy to use applications generated from its code.

Sun Microsystems will continue to own and protect the brand and its value by managing its use. Any product that wishes to use one of the family of Java brands must comply with the associated Java brand program. These are outlined at <http://www.java.com/en/about/brand/>

⏮ Back to top

Q: Can I call products I create from code I download from your open-source site "Java"?

A: If your product meets one of several program requirements for using the Java brand, including the applicable testing, (see <http://www.java.com/en/about/brand/>) then you can use the appropriate Java cup and steam logo, according to the guidelines of the brand.

For example:

- "My product, Foo Sneakers is Java Compatible" (add logo)
- "This derivative of the OpenJDK project source code is Java Compatible" (logo)
- "I created my application using a Java Compatible implementation and it is Java Powered" (logo)

The Java Powered logo requires that you be a member of the Sun Partner Advantage Program and ship a Java-based application for Java SE 1.4 or later. The Java Compatible logo program for Java SE requires that your implementation passes the relevant TCs for Java SE, and that you apply for the logo. If however, you choose not to pursue and meet the requirements of a Java brand program, then you are not granted rights to use the Java name or its logo. Instead, your use of the word Java is limited only to what is legally referred to as "fair use" (See question below). Please note that under US law, and other jurisdictions, there is no "fair use" of a logo, and you must have a license.

For example, you can say:

- My product uses code I downloaded from the openjdk.dev.java.net website
- My product, JCool for Java Technology, can be used with Java SE 6

⏮ Back to top

Q: Can I use the term "JDK" to describe my product - It's built with code from the OpenJDK Project after all?

A: The trademark "JDK" is Sun's name for the Java Development Kit. Accordingly, it follows similar rules to the "Java" trademark. You may not refer to your product or implementation as a JDK, even if it is based on code you downloaded from the OpenJDK Project. You can use the term "JDK" under Sun's "fair use" guidelines, however. Please refer to Sun's trademark policy for more information.

⏮ Back to top

Q: How do I know if someone else's product is compatible?

A: If a product carries the Java cup and steam logo, and/or uses the term Java Compatible or Java Powered, then it is bound by Sun's trademark and compatibility program rules, and carries the guarantee of a tested, compatible product.

If a product describes a relationship to an open-source Java community and its code, such as "derived from code found at [openjdk dev java net](#)", then it is important you look further into its claims, and ask for more concrete evidence that the code has been tested and certified as "Java powered" or "Java Compatible." For instance, you might ask if the solution has been certified by the appropriate TCK test suite.


⌂ Back to top

Q: How can I describe a product I create using code from the open-source site if I don't wish to get it certified by Sun?

A: If you create a derivative product from code you download from one of the open-source Java technology project sites on [java.net](#) such as [openjdk dev java net](#), it is not considered "Java Compatible" until it is certified, it is instead, a "derivative work based on the open-source code from the OpenJDK Project." While it may truly be a derivative of Java technology, trademark law only supports use of the word if it's verifiable and licensed by Sun.

Trademark law supports however, a notion of "fair use" that allows the use of a trademarked word as text under certain guidelines.

When making fair use of a Java trademark, you should acknowledge that Sun Microsystems owns the trademark. The following language is appropriate:

"Java, JDK, OpenJDK, Java Compatible, Java Powered, and  are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. or other countries."

For a comprehensive description of how and when you are allowed to use the Java trademark, please refer to <http://www.sun.com/policies/trademarks/>

⌂ Back to top

Q: What is the Java Verified Program?

A: The Java Verified Program is a wireless industry driven organization that provides a complete testing process for mobile Java technology applications. The member companies consisting of Motorola, Nokia, Siemens, Sony Ericsson, Vodafone Group, LG Electronics, Orange and Sun Microsystems create the testing criteria by which the mobile applications are tested. Those applications that pass the certification testing gain the right to use the Java Powered Logo and a digital certificate to prove the authenticity and integrity of the software.

→ TCKs

- What are TCKs?
- Will developers have the ability to run TCK tests on components they modify?
- What is the procedure for an individual or organization to apply for and receive a TCK support scholarship from Sun through the JCP?

Q: What are TCKs?

A: Technology Compatibility Kits (TCKs) are the compliance tests, tools and documentation which allows you to establish whether a particular implementation completely and correctly implements a particular Java technology specification as defined by a Java Specification Request (JSR) on a specific host platform. If your product or implementation passes the TCK for a specification, and meets all the untested compatibility requirements for a specification described in its TCK, it compatibly implements that specification.

• Back to top

Q: Will developers have the ability to run TCK tests on components they modify?

A: We know that access to the TCK tests is important to developers working on the open-source code base. However, we've been completely focused on the mechanics of getting the initial source code release completed, and at this time do not have any information to provide on TCK access.

For some time Sun has offered the Compatibility Testing Scholarship Program as a way for qualified educational institutions and non-profits to gain access to TCK's and support services. The TCK Scholarship program will continue.

• Back to top

Q: What is the procedure for an individual or organization to apply for and receive a TCK support scholarship from Sun, through the JCP?

A: Please see the section "How to apply for a scholarship" on the website.

JCP

- Will open-sourcing Java SE and Java ME require any changes in the JCP?
- Where will Java SE and Java ME specification evolution occur?
- How can I influence the evolution of Java SE or Java ME?
- How do I join the JCP?
- My employer won't allow me to join the JCP as an individual. I still would like to participate in Java's evolution. so what can I do?

Q: Will open-sourcing Java SE and Java ME require any changes in the JCP?

A: No. Many open-source implementations of JSRs developed under the JCP are available today.

For example, Sun's reference implementation of JSR 244, Java EE 5, has already been open sourced as the GlassFish application server.

Nonetheless, the JCP itself is constantly reviewing its own rules to ensure they are fair and modern and indeed, is currently engaged in revisions - see below.

⌚ Back to top

Q: Where will Java SE and Java ME specification evolution occur?

A: The JCP defines the process whereby Java technology specifications are created and updated. This will not change. Today, Sun is the Spec Lead for the umbrella JSRs which define the components of Java SE. The Java SE Expert Group works with the Spec Lead to determine which additional JSRs and other minor improvements will be included in each new release of the specification. Final approval for all changes is made by a vote of the JCP Executive Committee. The same process holds true for the Java ME platform.

⌚ Back to top

Q: How can I influence the evolution of Java SE or Java ME?

A: There are several ways to influence the direction of Java SE or Java ME platforms: 1) you can join the JCP and propose or join a JSR which will be included in Java SE or Java ME specifications, 2) comment on such JSR specs during public reviews, 3) work with Sun or a member of a JSR Expert Group to sponsor your ideas, 4) work with Sun in the OpenJDK or Mobile&Embedded communities to develop minor enhancements to the implementation (i.e., small improvements not warranting a separate JSR). The JCP Program Office has recently sponsored JSR 306 which in part will look into ways to get more individual involvement in JSRs.

⌚ Back to top

Q: How do I join the JCP?

A: Join the JCP by signing the Java Specification Participation Agreement (JSPA). You can find the JSPA and instructions on how to fill it out and send it in here: <http://jcp.org/en/participation/membership>.

⌚ Back to top

Q: My employer won't allow me to join the JCP as an individual. I still would like to participate in Java's evolution, so what can I do?

A: Many employers have policies about employee involvement with outside groups such as the JCP or other standards bodies. There are often good reasons for such policies and your employer should be able to explain them to you. As described above, however, there are ways to participate without signing the JSPA. In addition, one of the goals for JSR 306 is to look for means to increase participation of individuals and to allow more outside influence into the JSR development process.

Still, as a non-member you have many options to participate and influence: all spec drafts are publicly accessible, and you can provide direct feedback at any time to the spec lead and expert group by emailing the comments (which you can find on each JSR's web page).

Last but not least, in order to stay informed about the Java Community Process, you may subscribe to the JCP mailing list JCP-INTEREST. Visit the web site for more details.

This list will allow you to follow the progress of specification proposals as they go through the Community Process. You will receive messages when new specifications are proposed, when revisions to existing specifications are requested, when specification drafts are available, and when the specifications are finalized.

Sun Opens Java

Share the innovation.



[Overview](#) [Java Story](#) [What They're Saying](#) [What Universities Are Saying](#)

[» Contact Me](#)

13 November 2006—Sun believes deeply in creating communities and sharing innovations and technologies to foster more participation. Today in a historic move, Sun is opening the door to greater innovation by open sourcing key Java implementations—Java Platform Standard Edition (Java SE), Java Platform Micro Edition (Java ME), and Java Platform Enterprise Edition (Java EE)—under the GNU General Public License version 2 (GPLv2), the same license as GNU/Linux.

Sun is now the biggest contributor to the open-source community. Already Sun has released open-source implementations of its Solaris Operating System, NetBeans, Project Looking Glass, Project JXTA, Jini, OpenOffice, OpenSPARC, and Java EE technologies and is continuing on its path to open all of its middleware. By adding a second GPLv2 license to Java EE, which was previously available under the CDDL license through Project Glassfish, Sun is now open sourcing all core Java technologies under the same license.

Through this move, Sun is helping to

- Fuel innovation and build broader developer communities by enabling interested Java developers, as well as developers in the GNU/Linux community, to contribute more easily to the evolution of Java technology
- Drive faster evolution of the Java platform and adoption by new developers and in new markets while ensuring platform quality and flexibility
- Allow the 5 million Java developers worldwide to leverage platform enhancements and speed time to market for new applications

Enlisting the World to Innovate

For the past 11 years, Java technology has enabled developers to Write Once, Run Anywhere. Sun's commitment to compatibility and choice has made Java the most widely deployed application platform. Java technology is currently used on more than 4 billion devices worldwide, and the Java ME platform ships on more than eight of every 10 mobile handsets.

Sun believes Java technology has reached the right level of maturity, adoption, and innovation—with widespread use across enterprises and devices—to move into the next stage of its evolution. In the largest single contribution under the GNU GPL, Sun is releasing all of its key Java implementations under this widely respected free-software license.

- **Open-Source Java SE:** Today Sun is releasing the source code for the Java HotSpot virtual machine, the Java programming language compiler (javac), and JavaHelp online help software. Release of a fully buildable Java SE Development Kit (JDK) based nearly entirely on open-source code is expected in the first half of 2007.
- **Open-Source Java ME:** Sun is first releasing the source code for Sun's Java ME Feature Phone implementation based on Connected Limited Device Configuration (CLDC), which currently enables rich mobile data services in more than 1.5 billion handsets, and the source code for the Java ME testing and

Get the Source Code

[OpenJDK](#)
[Mobile & Embedded](#)
[GlassFish](#)

Get Involved

Participate in the new open source Java communities OpenJDK, Mobile & Embedded and the GlassFish community.

Related

- » [Open Source Java](#)
- » [James Gosling's Letter to the community](#)
- » [Press Kit](#)
- » [Jonathan's Blog](#)
- » [James Gosling's Blog](#)
- » [Rich Green's Blog](#)
- » [Simon Phipps' Blog](#)
- » [Hal Stern's Blog](#)

compatibility kit (TCK) framework. Later this year, Sun will release additional source code for the Advanced Operating System Phone implementation for based on the Connected Device Configuration (CDC) specification and the framework for the Java Device Test Suite.

- **New Developer Communities:** Tapping its experience in building dynamic and transparent open-source communities, Sun is launching the OpenJDK Community and the Mobile & Embedded Community to support developer participation in evolving the open-source JDK and open-source Java ME implementations respectively.

Developers wanting to get started right away can take advantage of the best tool for open-source Java application development: the NetBeans Integrated Development Environment (IDE). The NetBeans IDE provides complete support for the entire Java platform, from Java ME to Java SE to Java EE. To further speed time to market, Sun is also providing pre-built NetBeans projects at netbeans.org for the Java language components being open sourced and is making the Sun Studio development environment available at developers.sun.com/sunstudio for the native Java language components.

Open-Source Opportunities

By open sourcing its Java implementations, Sun will open new market opportunities, fuel innovation, and drive broader adoption of this Web 2.0 platform while minimizing fragmentation in the mobile community by delivering a consistent application platform across devices.

- Developers and ISVs can build differentiated Java technology-based applications and value-added Web 2.0 services more quickly through access to the latest Java source code.
- Developers can improve platform quality and functionality by contributing feature enhancements, bug fixes, and testing results to the open-source Java initiatives.
- Customers can lower costs and protect technology investments by taking full advantage of open-source business models that allow for free substitution of alternative operating systems, architectures, middleware, and devices on industry-standard hardware.
- Governments and educational institutions can reap the benefits of open-source Java technologies while ensuring security, privacy, and datacenter control.
- GNU/Linux distributors can add no-cost Java implementations to their distributions, while customers with stringent open-source requirements can deploy a free, reliable Java software stack on most GNU/Linux distributions.

Sun is taking careful and deliberate action as it open sources its Java technology implementations to help ensure that Java remains a central unifying standard for the Internet. Whether developers and customers choose to use Sun's commercial Java platforms or new open-source implementations, Java technology will continue to deliver the compatibility, stability, and quality required to turn Web 2.0 advancements into competitive advantage.

Go to sun.com/opensource/java for more information.

Virtual machine

From Wikipedia, the free encyclopedia

In computer science, a **virtual machine** is software that creates a virtualized environment between the computer platform and its operating system, so that the end user can operate software on an abstract machine.

Contents

- 1 Definitions
 - 1.1 Hardware virtual machine
 - 1.2 Application virtual machine
 - 1.3 Virtual environment
 - 1.4 Machine aggregation
- 2 Techniques
 - 2.1 Emulation of the underlying raw hardware (native execution)
 - 2.2 Emulation of a non-native system
 - 2.3 Operating system-level virtualization
- 3 List of hardware with virtual machine support
- 4 List of virtual machine software
 - 4.1 Extended descriptions of selected virtualization software
- 5 Books
- 6 See also
- 7 External links

Definitions

Specifically, the term virtual machine has several distinct meanings:

Hardware virtual machine

See also: Virtualization and Comparison of virtual machines

The original meaning of **virtual machine**, sometimes called a **hardware virtual machine**, is that of a number of discrete identical execution environments on a single computer, each of which runs an operating system. This can allow applications written for one OS to be executed on a machine which runs a different OS, or provide execution "sandboxes" which provide a greater level of isolation between processes than is achieved when running multiple processes on the same instance of an OS. One use is to provide multiple users the illusion of having an entire computer, one that is their "private" machine, isolated from other users, all on a single physical machine. Another advantage is that booting and restarting a virtual machine can be much faster than with a physical machine, since it may be possible to skip tasks such as hardware initialization.[1]

Such software is now often referred to with the terms virtualization and virtual servers. The host software which provides this capability is often referred to as a **virtual machine monitor** or **hypervisor**.

Software virtualization can be done in four major ways:

- Emulation, full system simulation, or "full virtualization with dynamic recompilation" -- the virtual machine simulates the complete hardware, allowing an unmodified OS for a completely different CPU to be run.
- Paravirtualization — the virtual machine does not simulate hardware but instead offers a special API that requires OS modifications.
- Native virtualization and "full virtualization" — the virtual machine only partially simulates enough hardware to allow an unmodified OS to be run in isolation, but the guest OS must be designed for the same type of CPU. The term *native virtualization* is also sometimes used to designate that hardware assistance through Virtualization Technology is used.

Application virtual machine

Another meaning of **virtual machine** is a piece of computer software that isolates the application being used by the user from the computer. Because versions of the virtual machine are written for various computer platforms, any application written for the virtual machine can be operated on any of the platforms, instead of having to produce separate versions of the application for each computer and operating system. The application is run on the computer using an interpreter or Just In Time compilation. One of the best known examples of an application virtual machine is Sun Microsystem's Java Virtual Machine.

See also: Comparison of Application Virtual Machines.

Virtual environment

A **virtual environment** (otherwise referred to as Virtual private server) is another kind of a virtual machine. In fact, it is a virtualized environment for running user-level programs (i.e. not the operating system kernel and drivers, but applications). Virtual environments are created using the software implementing operating system-level virtualization approach, such as Virtuozzo, FreeBSD Jails, Linux-VServer, Solaris Containers, and OpenVZ.

Machine aggregation

A less common use of the term is to refer to a computer cluster consisting of many computers that have been aggregated together as a larger and more powerful "virtual" machine. In this case, the software allows a single environment to be created spanning multiple computers, so that the end user appears to be using only one computer rather than several.

PVM (Parallel Virtual Machine) and MPI (Message Passing Interface) are two common software packages that permit a heterogeneous collection of Unix and/or Windows computers, hooked together by a network, to be used as a single, large, parallel computer. Thus large computational problems can be solved more cost effectively by using the aggregate power and memory of many computers.

The Plan9 Operating System from Bell Labs uses this approach.

Boston Circuits had released the gCore (grid-on-chip) Central Processing Unit (CPU) with 16 ARC 750D cores and a Time-machine hardware module to provide a virtual machine that uses this approach.

Techniques

Emulation of the underlying raw hardware (native execution)

This approach is described as full virtualization of the hardware, and can be implemented using a Type 1 or Type 2 hypervisor. (A Type 1 hypervisor runs directly on the hardware; a Type 2 hypervisor runs another operating system, such as Linux.) Each virtual machine can run any operating system supported by the underlying hardware. Users can thus run two or more different "guest" operating systems simultaneously, in separate "private" virtual computers.

The pioneer system using this concept was IBM's CP-40, the first (1967) version of IBM's CP/CMS (1967-1972) and the precursor to IBM's VM family (1972-present). With the VM architecture, most users run a relatively simple interactive computing single-user operating system, CMS, as a "guest" on top of the VM control program (VM-CP). This approach kept the CMS design simple, as if it were running alone; the control program quietly provides multitasking and resource management services "behind the scenes". In addition to CMS, VM users can run any of the other IBM operating systems, such as MVS or z/OS. z/VM is the current version of VM, and is used to support hundreds or thousands of virtual machines on a given mainframe. Some installations use Linux for zSeries to run Web servers, where Linux runs as the operating system within many virtual machines.

Full virtualization is particularly helpful in operating system development, when experimental new code can be run at the same time as older, more stable, versions, each in separate virtual machines. (The process can even be recursive: IBM debugged new versions of its virtual machine operating system, VM, in a virtual machine running under an older version of VM, and even used this technique to simulate new hardware.^[1])

The x86 processor architecture as used in modern PCs does not actually meet the Popek and Goldberg virtualization requirements. Notably, there is no execution mode where all sensitive machine instructions always trap, which would allow per-instruction virtualization.

Despite these limitations, several software packages have managed to provide virtualization on the x86 architecture, even though dynamic recompilation of privileged code, as first implemented by VMware, incurs some performance overhead as compared to a VM running on a natively virtualizable architecture such as the IBM System/370 or Motorola MC68020. By now, several other software packages such as Virtual PC, VirtualBox, Parallels Workstation and Virtual Iron manage to implement virtualization on x86 hardware.

On the other hand, plex86 can run only Linux under Linux using a specific patched kernel. It does not emulate a processor, but uses bochs for emulation of motherboard devices.

By now, Intel and AMD have introduced features to their x86 processors to enable virtualization in hardware.

Emulation of a non-native system

Virtual machines can also perform the role of an emulator, allowing software applications and operating systems written for another computer processor architecture to be run.

Some virtual machines emulate hardware that only exists as a detailed specification. For example:

- One of the first was the p-Code machine specification, which allowed programmers to write Pascal programs that would run on any computer running virtual machine software that correctly implemented the specification.
- The specification of the Java virtual machine.
- The Common Language Infrastructure virtual machine at the heart of the Microsoft .NET initiative.

This technique allows diverse computers to run any software written to that specification; only the virtual machine software itself must be written separately for each type of computer on which it runs.

Operating system-level virtualization

Operating System-level Virtualization is a server virtualization technology which virtualizes servers on an operating system (kernel) layer. It can be thought of as partitioning: a single physical server is sliced into multiple small partitions (otherwise called virtual environments (VE), virtual private servers (VPS), guests, zones etc); each such partition looks and feels like a real server, from the point of view of its users.

The operating system level architecture has low overhead that helps to maximize efficient use of server resources. The virtualization introduces only a negligible overhead and allows running hundreds of virtual private servers on a single physical server. In contrast, approaches such as virtualisation (like VMware) and paravirtualization (like Xen or UML) cannot achieve such level of density, due to overhead of running multiple kernels. From the other side, operating system-level virtualization does not allow running different operating systems (i.e. different kernels), although different libraries, distributions etc. are possible.

List of hardware with virtual machine support

- AMD Pacifica
- Boston Circuits gCore (grid-on-chip) with 16 ARC 750D cores and Time-machine hardware virtualization module.
- Freescale PowerPC MPC8572 and MPC8641D
- IBM System/370, System/390, and zSeries mainframes
- Intel Vanderpool

See also: X86 virtualization#Hardware support in x86 processors

List of virtual machine software

Application virtual machine software

- Common Language Runtime - C#, Visual Basic .NET, J#, Managed C++
- Corn concurrent runtime machine - Corn language
- EiffelStudio for the Eiffel programming language
- Erlang programming language
- Forth virtual machine - Forth
- Glulx - Glulx, Z-code
- Hec - HasM Assembler
- Inferno - Limbo
- Java virtual machine - Java, Nice, NetREXX
- Low Level Virtual Machine (LLVM) - currently C, C++, Stacker
- Lua
- Macromedia Flash Player - SWF
- MMIX - MMIXAL
- Neko virtual machine - currently Neko and haXe

Hardware virtual machine software

- ATL (A MTL Virtual Machine)
- Bochs, portable open source x86 and AMD64 PCs emulator
- CoLinux Open Source Linux inside Windows
- Denali, uses paravirtualization of x86 for running unmodified PC operating systems.
- FAUmachine
- LilyVM is a lightweight virtual machineAn introduction
- Microsoft Virtual PC and Microsoft Virtual Server
- Parallels Workstation, provides virtualization of x86 for running unmodified PC operating systems.
- QEMU, is a simulator based on a virtual machine.
- Simics
- SVISTA
- TRANGO real-time embedded hypervisor

- O-code machine - BCPL
- P-Code machine - Pascal
- Parrot - Perl 6
- Perl virtual machine - Perl
- Portable.NET - C#, Visual Basic .NET, J#, Managed C++
- YARV - Ruby
- ScummVM - Scumm
- SECD machine - ISWIM, Lispkit Lisp
- Sed the stream-editor can also be seen as a VM with 2 storage spaces.
- Smalltalk virtual machine - Smalltalk
- SQLite virtual machine - SQLite opcodes
- Squeak virtual machine - Squeak
- TrueType virtual machine - TrueType
- Valgrind - checking of memory accesses and leaks in x86/x86 64 code under Linux
- VX32 virtual machine - application-level virtualization for native code
- Virtual Processor (VP) from Tao Group (UK).
- Waba - Virtual machine for small devices, similar to Java
- Warren Abstract Machine - Prolog
- Z-machine - Z-Code
- Zend Engine - PHP
- twoOStwo
- User-mode Linux
- Virtual Iron (Virtual Iron 3.1)
- VM from IBM, the first true virtual machine system and still in wide use today.
- VMware (ESX Server, Fusion, Virtual Server, Workstation, Player and ACE)
- Xen
- KVM
- VirtualBox

OS-level virtualization software

- OpenVZ
- Virtuozzo
- FreeVPS
- Linux-VServer
- FreeBSD Jails
- Solaris Containers

Extended descriptions of selected virtualization software

The following software products are able to virtualize the hardware so that several operating systems can share it.

- Adeos is a Hardware Abstraction Layer that can be loaded as a Kernel Module in Linux. It allows the loading of a real-time kernel as a module, at the same time as Linux but with higher priority.
- Denali uses paravirtualisation to provide high-performance virtual machines on x86 computers. Denali's virtual machines support specialised minimal OSs for Internet services. The system can scale to thousands of virtual machines. Denali does not preserve the application binary interface (ABI), and so applications must be recompiled to run within a library operating system; in this sense it is similar to the Fixkernel.
- OpenVZ - Operating System-level server virtualization solution, built on Linux.
- Parallels provides virtualization of x86 for running unmodified PC operating systems. It also uses the technology of lightweight hypervisor in order to improve security and to increase the efficiency. Parallels has become popular for its ability to run Windows as a guest under Mac OS X on the Apple-Intel architecture.
- Qemu is a simulator based on a virtual machine, which gives it the ability to emulate a variety of guest CPU architectures on many different host platforms.
- Virtual Iron provides virtual machines for x86 that run unmodified operating systems, such as Windows, RedHat and Suse. Virtual Iron open source virtualization technology implements native virtualization, which delivers near-native performance for x86 operating systems.
- Virtuozzo replaces the hardware abstraction layer with a modified version enabling it to run with better performance of the OS, but forces all the VMs on a hardware box to all run the same OS, with some flexibility to support various Linux distributions on the same server. Currently they have a version for Windows 2003 and for Linux. OpenVZ is a related open-source project providing similar functionality for Linux.

- VMware provides virtual machines for x86 that can run unmodified PC operating systems. The technology involved in doing this is complex and also incurs (sometimes significant) performance overheads. Xen trades running of existing operating systems for running modified (paravirtualized) operating systems with improved performance. Virtual Iron provides full OS compatibility for existing or new OSes with near-native performance without the performance trade-offs between paravirtualization andn binary translation. VMware is by far the most popular VM for Windows virtualization.
- Xen Virtualization system whose motivation differs from that of Denali in that it is intended to run a moderate number of full-featured operating systems, rather than a large number of specialised, lightweight ones.
- KVM is a Linux kernel module that enables a modified Qemu program to use hardware virtualization. The latest CPU benchmarks show KVM is faster than Xen.^[2]

Books

- Jim, Jr. Smith, Ravi Nair, James E. Smith , Virtual Machines: Versatile Platforms For Systems And Processes, Publisher Morgan Kaufmann Publishers, May 2005, ISBN 1-55860-910-5, 656 pages

See also

- Comparison of virtual machines
- Comparison of Application Virtual Machines
- Virtual appliance
- Computing
- ICL's VME operating system
- Gerrit Blaauw (A significant designer of the IBM/360, an early virtual computing architecture)
- L1VM (The University of Illinois's Low Level Virtual Machine, a compiler toolset)
- Threaded code (A common implementation technique for application virtual machines)
- Virtual keyboard

External links

- The Reincarnation of Virtual Machines, Article on ACM Queue by Mendel Rosenblum, Co-Founder, VMware
- Citations from CiteSeer
- Performances comparison between Xen, UML, Vserver and VMware

Retrieved from "http://en.wikipedia.org/wiki/Virtual_machine"

Categories: [Virtualization software](#) | [Operating system technology](#) | [Programming language implementation](#)

- This page was last modified 15:06, 4 February 2007.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.